RESEARCH PLAN PROPOSAL

Performance Enhancement of Distributed Database Queries by Cache Investment

For registration to the degree of

Doctor of Philosophy

IN THE FACULTY OF SCIENCE

The IIS University, Jaipur

Under Supervision of:

Prof.(Dr.) Swati V. Chande

Principal MCA, ISIM

Submitted By:

Sanju Gupta

ICG/2011/13026

Department of Computer Science

July, 2012

1.0 INTRODUCTION

In recent years, with the development of computer network and database technology, distributed database is more and more widely used; with the expanding application, data queries are increasingly complex, the efficiency requests are increasingly high, so the performance enhancement of query processing is a key issue of the distributed database system.

Caching has emerged a fundamental technique for ensuring high performance in distributed database system.

1.1 DISTRIBUTED DATABASE OVERVIEW

A Relational Database is defined as a set of logically inter related data stored in the form of rows and columns. An organized set of programs are needed to handle the huge databases created by various organizations. A Database Management System is a software package that controls the creation, maintenance, storage, modification, and extraction of information from data records, files and other objects.

With the advancement of Computer Networks and increase in the size of the database, the decentralization of databases is in focus where information can be shared and retrieved from multiple locations. The computing environments are now composed of large number of workstations and mainframes interconnected to each other by high bandwidth communication networks. This has lead to the development of Distributed Database over multiple machines where the distribution of the database is Transparent to the users

Distributed database system is physically distributed and logically centralized database system, is the product generated from the mutual penetration and organic integration of computer network technology and database technology. Physical dispersion refers to that the data composing the distributed database is distributed to the different computers in the network, and each site in the network has the ability to deal with and can implement local applications. Concentrated in logic refers to that each site is a logical whole which is managed by a distributed database management system, and each site implements the global application through the network communication subsystem.

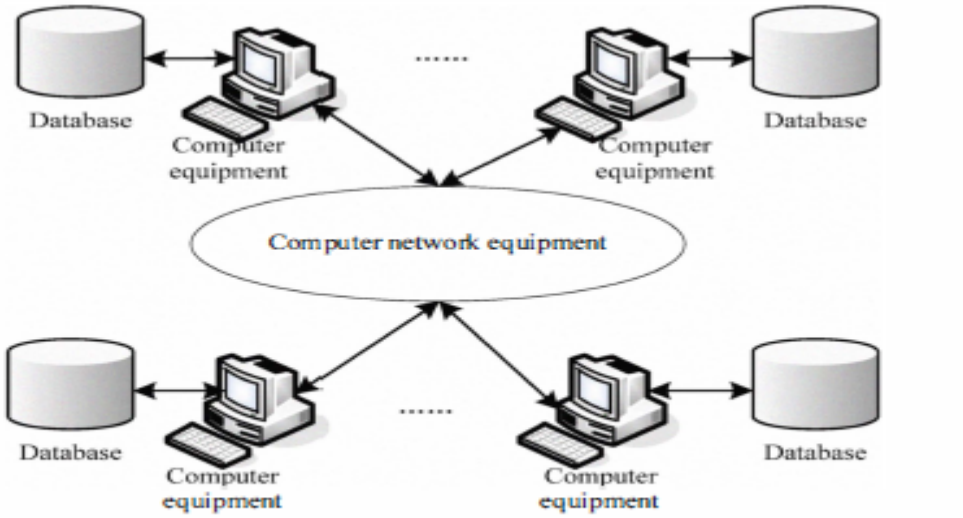


Figure 1. Distributed database system architecture

Figure1: Distributed database system architecture [9].

1.2 QUERY PROCESSING AND OPTIMIZATION

Query processing and optimization is very important in the relational database system, and also is one of the main research questions of the distributed database.

Query processing is the process of translating a query expressed in a high-level language such as SQL into low-level data manipulation operations. Query Optimization refers to the process by which the *best* execution strategy for a given query is found from a set of alternatives.

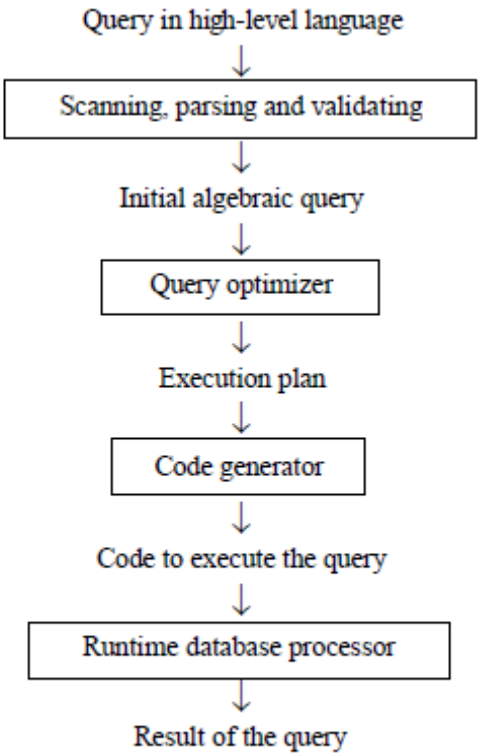


Figure 1. Query processing steps.

Figure 2: Query Processing Steps [23]

In distributed database systems, two different objectives are often used to consider the query optimization-

One objective uses the standard of minimum total cost, and in addition to considering CPU and I/O cost like the centralized database system, the total cost also includes the data network transmission cost . Which is because that the data distribution and redundancy make the communication cost needed by transmitting data among the sites in the query processing should be considered, which will cause the total cost increased.

Another objective uses the standard of minimum response time of each query, which has a great signification in the distributed database system. Which is because that the distributed database system is a system composed by multi computers, in which data distribution and redundancy has also increased the possibility of parallel processing, thus the response time of query processing can be reduced and the query processing speed will be speeded up.

1.3 CACHE INVESTMENT

Caching has emerged as a fundamental technique for ensuring high performance in distributed systems.

Caching is an opportunistic form of data replication in which copies of data that brought to a site by one query are retained at that site for possible use by subsequent queries.

Caching is particularly important in a large system with many clients and servers because it reduces communication costs and off-loads on shared sever machines.

Caching has been successfully integrated into many commercial and research database systems, data warehouse, and database application systems e.g. SAP R/3.

Cache Investment, is a novel technique for combining data placement and query optimization. Rather than requiring the creation of a new optimizer from scratch, Cache Investment is implemented as a module that sits outside the query optimizer. This module influences the optimizer to sometimes make suboptimal operator site selection for individual queries in order to effect a data placement that will be beneficial for subsequent queries. In other words, it causes the optimizer to invest resources during the execution of one query in order to benefit later queries.

There are two techniques for database caching. These are –

1) Physical Caching –Physical Caching is performed in terms of records, pages (blocks) or static partitions of base tables.
   Physical Caching is used by most distributed systems and database application.
2) Logical Caching-Logical Caching is performed in terms of query results or subsets of query results.

1.4 TYPES OF CACHE INVESTMENT POLICIES

Cache Investment Policies determine when and for which fragments the investment requires initiating caching. These policies are invoked for each query that is submitted at a client and can influence the way the operator site selection is done for that query.

These policies are-

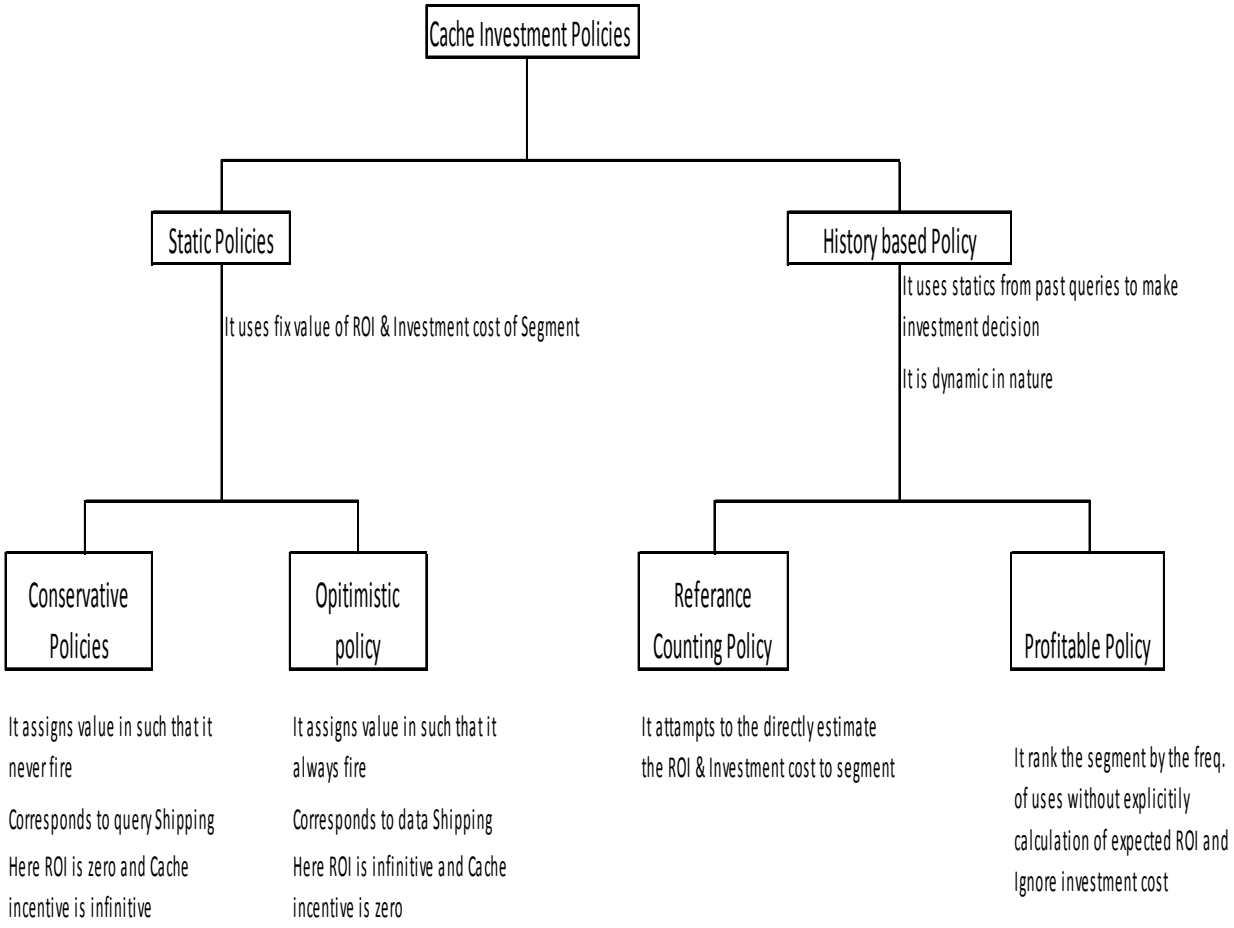

Cache Investment Policies
Static Policies
History based Policy
It uses fix value of ROI & Investment cost of Segment
It uses statics from past queries to make investment decision
It is dynamic in nature
Conservative Policies
Opitimistic policy
Referance Counting Policy
Profitable Policy
It assigns value in such that it never fire
Corresponds to query Shipping
Here ROI is zero and Cache incentive is infinitive
It assigns value in such that it always fire
Corresponds to data Shipping
Here ROI is infinitive and Cache incentive is zero
It attampts to the directly estimate the ROI & Investment cost to segment
It rank the segment by the freq. of uses without explicitily calculation of expected ROI and Ignore investment cost

Figure 3: Cache Investment Policies [18]

2.0 REVIEW OF LITERATURE

All relational database application programs are written in High Level Languages integrating relational language with a declarative interface like SQL to access the data stored in a database. It has been pointed out in [1], that the processing of distributed execution plan from the query via a computer network takes place in three phases: Local Processing Phase or Materialization Phase for all selections and projections, Reduction Phase for reducing the size of relations and cost of communications and Final Processing Phase or Transmission Phase to generate the output. An important objective in query optimization in distributed query is to reduce the amount of data transmission required and the cost of query execution for the second and third phase [2][3]. The work related to query optimization in databases started in late seventies which motivated a large part of database scientific community. The optimizer's role is to

generate an optimal execution plan from the considered search space. The optimization goal is to minimize response time and maximize throughput while minimizing optimization cost.

The cost of query execution over multiple locations in a Distributed environment is described as the sum of local cost (Input / Output cost, Cost of processor i.e. CPU cost at each site) and the cost of transferring data between sites. The complexity and cost increases with the increasing number of relations in the query. Further, minimizing the amount of data transmission is important to reduce the query processing cost. Due to the large number of parameters affecting query execution, a single query can be executed in several different ways.

The optimization process of a distributed query is composed of two components [5]: a) Global Optimization b) Local Optimization. The Global Optimization mainly consists of determining the best execution site for local sub query and finding the best inter-site operator scheduling. The Local Optimization is mainly focused on optimizing each local sub query on each site. A lot of research was also made on various factors like optimization algorithms, search space, execution strategies and cost model to optimize a query. Search Space is defined as the set of all Query Execution Plans that compute the same result. Every point of search space has cost associated with it. The Search Strategies refers to the enumeration algorithm for the optimizer to determine the best plan to execute the query and Cost Model allowing interpret operators' trees in the considered search space and estimating the cost of execution [4].

In [7], the author shows that the cost of evaluation can be reduced by exploiting common sub-expression and maintaining intermediate data structure. In this paper for efficient execution of the query in DDBMS the author designs the module in which they use two methods. 1) Cache investment and 2) Multi query optimization. The cache investment is used to determine which data items should be cached at the client. This improves a stream of queries submitted at a client. While multi query optimization is used to exploiting common sub expression. In this study efficiency of query is improved by reducing the evaluation cost, communication cost and response time.

In [8], the author has worked on the existing query optimization method to improve the performance of a query. He improves the performance of query by reducing the amount of middle data and reduces the total cost of network communication.

He redefines the existing query optimization process by adding a new user module and completes the data dictionary by adding a query sentence table in existing process. This sentence table is used to store the result of frequently used query, thus the transmission of large amount of data in query can be avoided and the query efficiency is improved.

In [6], the author reviews the traditional Static Optimization Strategies like SDD-1, R*, Dynamic Optimization Strategies and Randomized Strategies like Iterative Improvement Algorithms for non-autonomous distributed database systems and analyze the suitability of these strategies for autonomous systems. They further reviewed Mariposa, Query trading with DP and IDP and Query Trading with Processing task Trading (QTPT) for autonomous distributed database systems.

In [9], Fan & Me design a new algorithm to improve optimization efficiency. This new algorithm can significantly reduce the amount of intermediate result data and effectively reduce the network communication cost. New optimized algorithm is based on the Semi-connection operation. The basic principal of this optimization strategy is to use semi-connection operation in accordance with the connection order between relationships. The advantage of traditional semi-connection strategy is that it is simple to process but the deficiency is that it is not considered how to optimize the semi-connection order of the sub-query for further reduce the network communication cost. Authors improved the exiting strategy by using the intermediate results generated by the sub-query as a deceive factor of the network cost and designed a new semi-connection query algorithm.

In [10], this paper addresses the Distributed query optimization problems such as Local optimization of semi joins, Join sequence optimization, Relation semi join on broadcasting network, Semi join optimization for tee queries in detail.

They studied and compare the different query algorithm such as, optimizers of R*, Distributed- INGRES, Fragment and Replication Strategies (FRS), Partition and Replication Strategies (PRS) and Local Reduction algorithms (LR). They calculate the cost from all these algorithm and made comparison to find out which one is most optimal algorithm.

Based on comparison a new technique is designed to process with minimum quantity of inside data transfer. The proposed technique is used to determine which relations are to

be partitioned into fragments and where the fragments are to be sent for processing. This technique is efficient comparing to other techniques, as more than one relation is allowed to remain fragmented which exploits parallelism, while replicating the other relation to the sites of the fragmented relation. It reduces the communication cost and local processing cost this also improves the query response time.

In [11], this article the query optimization algorithm based on multi-relation semi join is put forward for the Shortage of common multi-relation semi joins query method. That is to improve the problem of how to further reduce the network cost by cost estimation and semi join. This article makes the data volume of intermediate result produced in the implementation for sub query as the decisive factor of network cost. The optimization benefit of common multi-relation semi join query optimization algorithm is pointed in this article, which means the relative benefit by comparison of multi-relation semi join query optimization algorithm and common multi-relation semi join query method. The function for the optimization benefit of multi-relation semi join query optimization algorithm is defined in this article as

μ = (the network cost for the implementation of multi-relation semi join query method subtract the network cost for the implementation of multi-relation semi join query optimization algorithm)/ the network cost from the implementation of common multi-relation semi join query method.

Author designs a new algorithm to apply to this situation that takes buffer zone of distributed database system as the final assembly station of intermediate result of query. The experiment testifies that query optimization algorithm based on multi-relation semi join obviously reduces the data volume of intermediate result, and effectively increases the total cost of network communication.

In [13], the author introduces Dynamic Programming (State-of-Art) Semi Join reducers in the modern client /middleware system to reduce the cost of communication, to generate effective plans with Semi Joins and to exploit all resources of a system. The author presented two variants of Dynamic Programming Algorithms, which differed in their implementation complexity, query profiles, run times and the quality of plans they produced, called Access Root Algorithm when integrated with the optimizer, reduced the size of the base tables using semi joins and Join Root Algorithm which applied semi joins at all query levels for further reductions.

In [18] the authors identify the circular dependency between caching and query optimization. This dependency is explained in detail here. The client-server architecture with hybrid shipping is used to study query optimization with data caching.

This paper presents four policies, which determine when and for which data (Table, Partition of table) Caching initiate. Authors gives the procedure for integration of cache module with optimizer without changing basic components of optimizer such as query optimizer's search strategies, the query engine or the buffer manager.
Relative performance of these four cache investment policies is studied. Simulation environment is set-up and a workload is designed which consist of a sequence of two-way join queries. The following two types of queries are used- 1) NoSel 2) HiSel. The relation participating in these queries are chosen using two different distributions 1) Uniform 2) Zipf.

Table: 1

	Nosel	HiSel
Uniform	Low investment, low ROI	High investment, low ROI
Zipf	Low investment, high ROI	High investment, high ROI

To evaluate the performance of these policies experiment is carried out by considering the following factor and simulation environment-

1) Communication Cost 2) Throughput, single sever 3) Throughput, heterogeneous server 4) Impact of Invalidations 5) CPU cost of query optimization. 6) Tuning the History based policies.

In the performance experiments reference-counting and profitable policies showed significantly better performance than static policies in many situations. It was not possible to identify a clear winner b/w the Reference-Counting and profitable policies for e.g. system with heterogeneous servers or system that are designed to process many update transactions should employ the profitable policy other system should employ the reference-counting policy because it is easier to implement and has significantly less overhead.

Paper [19] addresses the new challenges that arise when integrating caching with advance query processing system. This paper starts with identifying the relationship between caching and query optimization.

This paper shows how the query optimizer can be extended with cache investment, so that it produces good query execution plane and implicitly makes long term caching decision at the same time. Here experiment is made in query optimizer without changing basic components of it.

In this is paper client server architecture with hybrid shipping was used. They only concentrate on implementation of physical caching at granularity of data table and index pages. For caching the data only client's main memory is used (disk caching is not used).

This paper is also used the same approach as previous paper. Cache investment is integrated as a separate module. When query is executed, the traditional optimizer requests the current state of the cache, cache investment policy argument the answer with the table and indices that should be cached.

In effect cache investment lies to the optimizer. It patches the cache content information passed to the optimizer so that optimizer believe that all the data items that should be cached are already present in the cache. This misinformation leads the optimizer to consider scan operator for such tables and indices at the client, even if none of their pages are cache, so that relevant data items are significantly cached. Caching candidate is identifying by the cache investment policies .This is done by calculating investment cost and ROI. If ROI is higher than the investment for particular candidate then this candidate is cached. In this paper author only uses two cache investment policies, reference counting and profitable policies for calculating cache investment for base table and indices. They give the detail procedure for identifying cache candidate. In order to study the effectiveness of the cache investment policies in a real client server database system, they implemented and integrated them with a query optimizer for the SHORE Distributed database system. For this they specify these wokloads-1) POINTRAGE 2) HOTCOLD 3) UNIFORM. Performance experiment showed that history base policies perform better than static policies.

This [21] paper presents the state of art of query processing for distributed database and information system. In this paper author discussed various query processing

techniques developed for different types of distributed database such as client server, peer to peer, multi-tier and homogenous and heterogeneous system. For each category, paper presented and discussed the set of query processing techniques; these techniques include special join techniques, techniques to reduce communication cost and techniques to exploit caching replication of data. Both query optimization and caching are extensively described in this paper. This Paper describes the difference b/w replication and caching.

3.0 MOTIVATION/ JUSTIFICATION AND RELEVANCE

Based on the review of Literature, Query processing in a distributed database requires transfer of data from one computer to another through a communication network. Query at a given site might require data from remote sites. The performance of the distributed database query depends on how fast and efficiently data can be retrieved from multiple sites. Several factors impact the performance of distributed query processing. These factors include:

a) Selection of appropriate site when data is cached at multiple sites.
b) Order of Query Operations like selection, projection, join and transmission.
c) Selection of caching Method to accelerate the speedy execution of the queries.
d) Selection of Join Order to reduce the number of Search Spaces and to create the best Query Execution Plan.
e) To reduce the amount of Data Transmission over the network using caching and intermediate data and hence reduce the overall cost of Query Execution.

Research in the development of distributed database architecture has focused on allowing the flexible and dynamic placement of query operation at various sites in the network. In such a system query operation can be placed at sites in a way that minimize expected communication cost, execution time or other metrics.

Cache investment is a technique for combining data placement and query optimization in a manner that does not change the query optimizer directly. Cache Investment affects data placement by influencing the optimizer to make suboptimal choices regarding operator site selection. These suboptimal choices will be based on policies producing cached data placement beneficial for subsequent queries.

Cache Investment would also help truly distributed system such as System R* or Mariposa. There are many direction in which Caching can be extended. One important area for future work is caching of Index, Caching of result or sub queries. Integration of Cache Investment with Semantic Caching is another promising direction. Caching can be combined with techniques for global memory management.

4.0 OBJECTIVES

The main objectives to conduct the research are:

- To study the implementation of cache investment policies for improvement of the performance of database queries.
- To investigate the relative performance of existing cache investment policies using Memory Caching and Disk caching.
- To improve the use of Cache for query processing in a distributed database system by exploring and extending Cache Investment Policies.
- To suggest improved Memory and Disk Caching policies and calculate the relative performance of the improved Cache Investment policies for Caching in main memory and in disk in distributed environment.
- To extend query optimization process by Caching of sub queries and result of queries.
- To enhance the method and algorithms for Caching of Index in distributed database system.
- To evaluate the performance of distributed database queries after implementing the enhancements in cache investment.

5.0 PLAN OF WORK AND METHODOLOGY

For Performance Enhancement of Distributed Database Queries using cache Investment the most feasible approach is systematic literature review, detail study of query optimization methods and Cache investment strategies, design a new module or modify

existing algorithms, Implementation of the proposed methods/algorithms and compare the improvements and benefits over existing methods/algorithms.

5.1 RESEARCH DESIGN

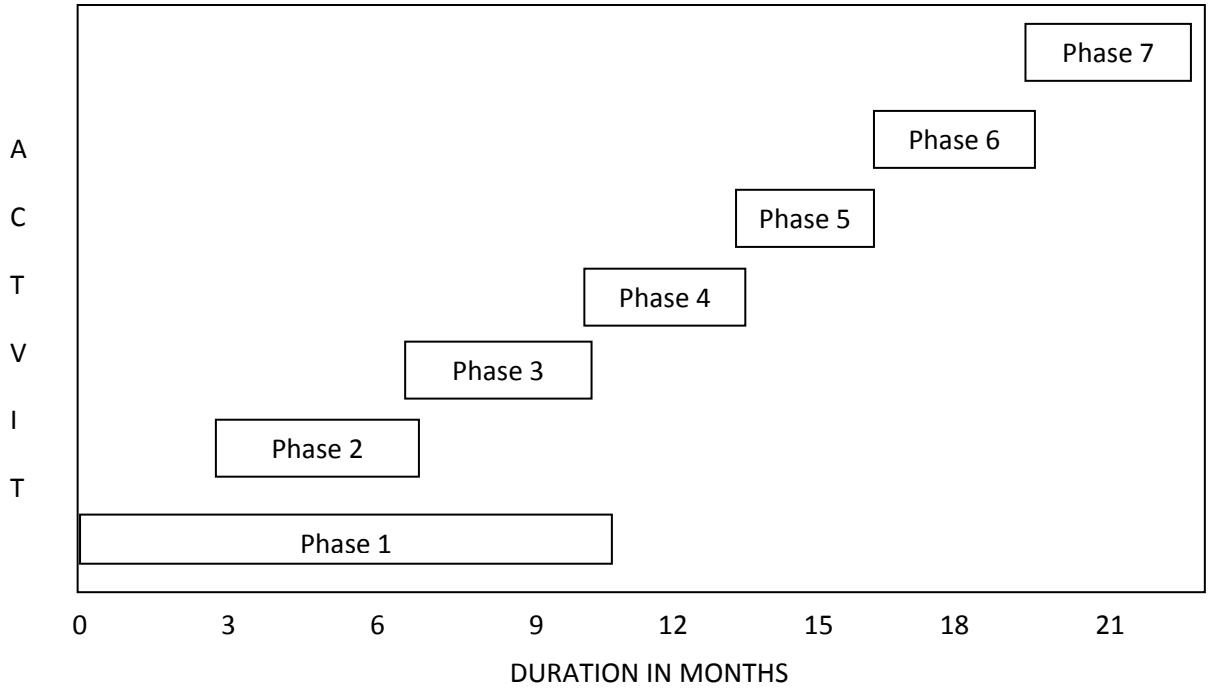
The query experiment of Distributed Database shall be created consisting of four to five work stations interconnected to each other via communication network. The Operating System used for the development shall be Linux / Windows. One of the work stations shall be converted into a Server. The Front End Development Environment and a Back End Relational Database Management System shall be installed on all the work stations.

5.2 WORK PLAN

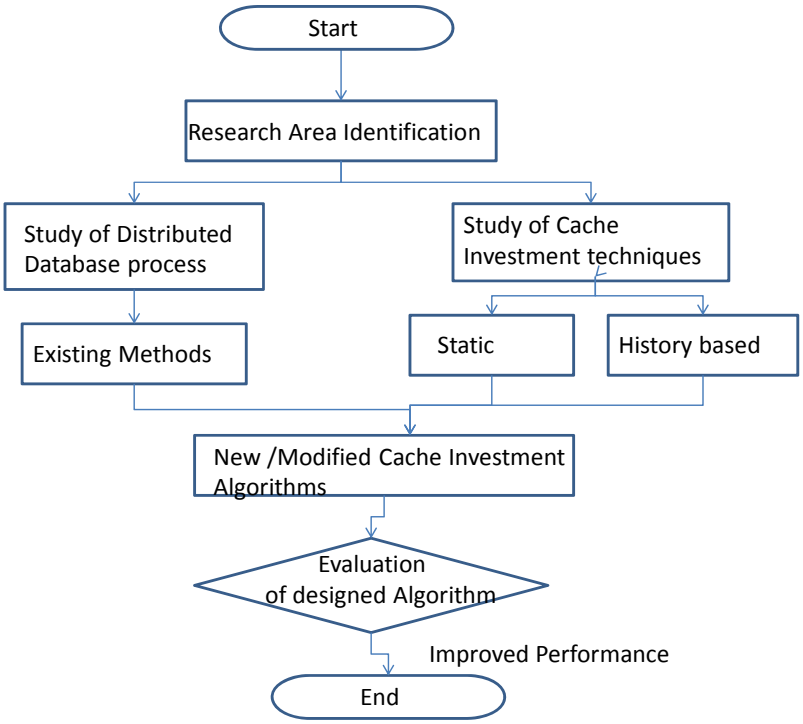
The research plan is divided into following seven phases:

Phase	Activity	Approx time(in months)
1.	Literature survey on Distributed database	8-12
2.	Detail study of query optimization methods, Cache Investment strategies and Algorithms	3-6
3	Assessment of drawn inferences from the proposed study	3-4
4.	Searching empirical databases	2-3
5.	Develop a new method or modify existing algorithms for improvement in performance of Query processing	3-4
6.	Implementation, comparison with existing algorithms and drawing conclusions	4-5
7.	Paper/ Thesis Writing	2-4

The timeline chart for research plan is as follow:



5.3 METHODOLOGY



5.4 TOOLS AND TECHNIQUES

Different relational databases use different types of tools, functions and commands to estimate the time taken to execute a query and hence to measure the performance of the query. Some tools are-1) SQL Profiler Tool is used in SQL Server 2008 to measure the performance of query. 2) T-SQL scripts using SSMS Query Editor in Test Environments can be used to find out the SQL Execution time in milliseconds .3) EXPLAIN [ANALYZE] [VERBOSE] statement is used in PostgreSQL to estimate the total elapsed time expended within each plan node in millisecond and also displays the total number of rows actually returned as display. 4) SetQueryTimeout() function or MYSQLCC is applied in mysql to estimate the Query Execution Time.

Various cost functions are also available with the query optimizer of the relational databases to estimate the cost incurred for the execution of the query. In distributed database environment, the additional cost of transferring the data over the communication channel is also important. The cost involved shall be estimates using the cost function of the given database. For example EXPLAIN PLAN Statement is used in Oracle to estimate the cost of the query execution.

6.0 PLACE OF WORK AND FACILITIES AVAILABLE

The experimental setup for the Distributed Database Environment shall be set up in the Information Technology Lab in The IIS University Campus.

7.0 REFERENCES

[1] C.T. Yu and C.C. Chang, “Distributed Query Processing” ACM Computational Surveys, vol. 16, Dec. 1984

[2] P.M.G. Apers, A.R. Hevner and S.B Yao, “Optimization Algorithm for Distributed Queries”, IEEE Trans. On Software Eng., Jan. 1983

[3] P.A. Bernstein, N.Goodman, E.Wong, C. Reeve and J.B. Rothnie, “ Query Processing in a System for Distributed Database(SDD-1)”, ACM Trans. Database Sys, Vol.6, Dec. 1981

[4] Abdelkader Hameurlain and Frank Morvan, “Evolution of Query Optimization Methods”, Trans on Large Scale Data and Knowl. Cent. Sys, 2009

[5] Ozsu M.T. and Valdureiz P: “Principles of Distributed Database System”, 2nd Edition, Prentice Hall, 1999.

[6] Doshi P. and Raisinghani V., “Review of Dynamic Optimization Strategies in Distributed Database”, Electronics Computer Technology (ICECT), 3rd International Conference, April 2011

[7]Ideh Azari, ”Efficient Execution of Query in Distributed Database System”, 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE),2010

[8]XUE Lin “Query Optimization Strategies and Implementation Based on Distributed Database” ,IEEE,2009.

[9] Fan Yuanyuan and Mi Xifeng,“Distributed Database system Query Optimization Algorithm Research,”IEEE,2010.

[10]R.M Monjurul Alom, Frans Henskens and Michael Hannaford, ” Query processing and optimization in Distributed database system,International Journal of Computer Science and Network Security(IJCSNS),Sep2009.

[11]Xianofeng Li,Dong Li,Hong Zhi Gao and Lu Yao,” : Study of Query of Distributed Database Based on Relation Semi Join”,International Conference On Computer Design and Application(ICCDA),2010.

[12] S.Pramanik, D.Vineyard. “Optimization Join Queries in Distribute Database”, IEEE , 2004.

[13] Konrad Stocker, Donald Kossmann, Reinhard Braumand and Alfons Kemper, “Integrating Semi-Join-Reducers into State-of-the-Art Query Processors”, Proceedings of the 17th International Conference on Data Engineering, HYPERLINK IEEE Computer Society Washington, DC, USA 2001.

[14] C.M.Chen, N.Roussopoulos, “Adaptive Selectivity Estimation Using Query Feedback, SIGMOD”, 2000.

[15] Shahabi C, Zarkesh A M, Adibi J, “Introduction of distributed database”. IEEE ,2001.

[16] Yan T,IacobesnM,Garcia-Mo Lina H,”Introduction of Query optimization of distributed database”. WAM Press, I 999.

[17] Xu L. “The reliability of distributed database”. Alaska:Anchorage Press,2001.

[18] Donald Kossmann , Michael J. Franklin. “Cache Investment Strategies". Univ.of MD Technical CS-TR-3803 and UMIACS-TR -97-50,May 1997.

[19] Donald Kossmann , Michael J. Franklin,Gehard Drasch, "Cache Investment : Integrating Query Optimization and Distributed Data Placement," ACM Transaction on Database System (TODS), Dec. 2000.

[20] Elmasri R. and Navathe S. B., *Fundamentals of Database Systems*, Reading, MA, Addison-Wesley, 2000.

[21] Donald Kossmann ," The State of the Art in Distributed Query Processing",ACM Computing Surveys,Dec.2000.

[22] Donald Kossmann , Michael J. Franklin,Gehard Drach,"Cache Investment for Indexes",VLDB Conference,Feb,1998.

[23]Alaa Aljanby ,Emad Abuuuuelrub and Mohammed odeh,"A Survey of Distributed Query Optimization", The International Arab Journal of Information Technology ,Jan 2005.